AMENDMENTS TO THE CLAIMS

Claims 1-15, 18-44, 47-70, 73 are pending in the instant application. Claim 73 has been added. The Applicant respectfully submits that the claims define patentable subject matter.

Listing of claims:

1. (original) A method for data verification, comprising:

receiving an input block of data together with a modulo-based input error detection code associated with the input block, the input block comprising a plurality of sub-blocks;

selecting a subset of the sub-blocks to be included in an output block;

determining an error correction term based on the selected subset; and

concatenating the selected subset of the sub-blocks together with the input error detection code and the error correction term to generate an output block for conveyance to a destination processor.

2. (original) The method according to claim 1, wherein the error correction term is equal to a binary difference between the input error detection code and an output error detection code of the output block.

3. (original) The method according to claim 1, wherein selecting the subset comprises determining an order of the sub-blocks in the output block, and wherein determining the error correction term comprises determining the error

correction term responsively to the order.

4. (original) The method according to claim 1, and comprising, upon receipt of the output block at the destination processor, determining whether to accept or reject the output block by computing an output error detection code of the output block, and comparing the output error detection code to the input error detection code and the error correction term.

5. (original) The method according to claim 4, wherein comparing the output error detection code comprises applying an XOR operation to the input error detection code and the error correction term.

6. (original) The method according to claim 1, wherein determining the error correction term comprises processing data in each of the sub-blocks so as to compute respective sub-block error detection codes.

7. (original) The method according to claim 6, wherein processing the data in each of the sub-blocks comprises taking a modulo of the data.

8. (original) The method according to claim 7, wherein taking the modulo comprises computing the modulo with respect to a predetermined polynomial, so as to determine a cyclic redundancy code (CRC) of the sub-block.

9. (original) The method according to claim 8, wherein computing the

modulo comprises using the predetermined polynomial that was applied in computing the input error detection code.

10. (original) The method according to claim 6, wherein the sub-blocks have respective input offsets within the input block, and wherein determining the error correction term comprises:

determining respective sub-block error correction terms for the sub-blocks, responsively to the respective sub-block error detection codes and input offsets; and

combining the sub-block error correction terms to determine the error correction term.

11. (original) The method according to claim 10, wherein determining the respective sub-block error correction terms comprises binary-shifting the respective sub-block error detection codes by respective numbers of bits equal to the respective input offsets, and computing respective modulos of the respective shifted values.

12. (original) The method according to claim 10, wherein the sub-blocks in the selected subset have respective output offsets within the output block, and wherein determining the respective sub-block error correction terms for the sub-blocks in the subset comprises determining the respective sub-block error correction terms responsively to the respective sub-block error detection codes, input offsets, and output offsets.

13. (original) The method according to claim 12, wherein determining the respective sub-block error corrections terms comprises multiplying the respective sub-block error detections codes by respective sums of (a) 2 raised to a power of a value of the respective input offsets and (b) 2 raised to a power of a value of the respective output offsets, and computing respective modulos of the respective multiplied values.

14. (original) The method according to claim 1, wherein the sub-blocks have respective input offsets within the input block, and wherein determining the error correction term comprises:

determining respective sub-block error correction terms for the sub-blocks by binary-shifting a value of each of the sub-blocks responsively to the respective input offsets, and computing respective modulos of the respective binary-shifted values; and

combining the sub-block error correction terms to determine the error correction term.

15. (original) The method according to claim 14, wherein the sub-blocks in the selected subset have respective output offsets within the output block, and wherein determining the respective sub-block error correction terms for the sub-blocks in the subset comprises binary-shifting a value of each of the sub-blocks responsively to the respective output offsets, and computing respective modulos of the respective binary-shifted values.

16. (withdrawn) A method for error detection, comprising: receiving a

block of data having a modulo-based input error detection code and an error correction term appended thereto;

calculating an output error detection code of the block;

combining the input error detection code and the error correction term to produce a modified error detection code; and

comparing the calculated error detection code to the modified error detection code so as to detect an error in the block.

17. (withdrawn) The method according to claim 16, wherein combining the appended error detection code and the error correction term comprises applying an XOR operation.

18. (original) A method for data processing, comprising:

receiving an input block of data together with a modulo-based input error detection code associated with the input block, the input block comprising a plurality of input sub-blocks;

generating one or more protocol-related sub-blocks to be incorporated together with the input sub-blocks in a specified order in an output block for conveyance to a destination processor;

determining an error correction term based on the specified order of the protocol-related sub-blocks and the input sub-blocks;

determining a modulo-based output error detection code responsively to the input error detection code and the error correction term; and

concatenating the protocol-related sub-blocks, the input sub-blocks and the output error detection code to generate an output block for conveyance to a

destination processor.

19. (original) The method according to claim 18, wherein the error correction term is equal to a binary difference between the input error detection code and the output error detection code.

20. (original) The method according to claim 18, wherein at least one of the protocol-related sub-blocks comprises a header, a marker, or padding.

21. (original) The method according to claim 18, wherein determining the output error detection code comprises applying an XOR operation to the input error detection code and the error correction term.

22. (original) The method according to claim 18, wherein determining the error correction term comprises processing the data in each of the protocol-related sub-blocks and the input sub-blocks so as to compute respective output sub-block error detection codes.

23. (original) The method according to claim 22, wherein processing the data in each of the protocol-related sub-blocks and the input sub-blocks comprises taking a modulo of the data.

24. (original) The method according to claim 23, wherein taking the modulo comprises computing the modulo with respect to a predetermined polynomial, so as to determine a cyclic redundancy code (CRC).

25. (original) The method according to claim 24, wherein computing the modulo comprises using the predetermined polynomial that was applied in computing the input error detection code.

26. (original) The method according to claim 22, wherein determining the error correction term comprises:

determining respective output offsets of the protocol-related sub-blocks and the input sub-blocks within the output block, responsively to the specified order;

determining respective sub-block error correction terms for the protocol-related sub-blocks and the input sub-blocks, responsively to the respective sub-block error detection codes and output offsets; and

combining the sub-block error correction terms to determine the error correction term.

27. (original) The method according to claim 26, wherein determining the respective sub-block error correction terms comprises binary-shifting the respective sub-block error detection codes by respective numbers of bits equal to the respective output offsets, and computing respective modulos of the respective shifted values.

28. (original) The method according to claim 26, wherein the input sub-blocks have respective input offsets within the input block, and wherein determining the respective sub-block error correction terms for the input sub-blocks comprises determining the respective sub-block error correction terms

responsively to the respective sub-block error detection codes, input offsets, and output offsets.

29. (original) The method according to claim 28, wherein determining the respective sub-block error corrections terms comprises multiplying the respective sub-block error detections codes by respective sums of (a) 2 raised to a power of a value of the respective input offsets and (b) 2 raised to a power of a value of the respective output offsets, and computing respective modulos of the respective multiplied values.

30. (original) The method according to claim 18, wherein determining the error correction term comprises:

determining respective output offsets of the protocol-related sub-blocks and the input sub-blocks within the output block, responsively to the specified order;

determining respective sub-block error correction terms for the protocol-related sub-blocks and the input sub-blocks by binary-shifting a value of each of the protocol-related sub-blocks and the input sub-blocks responsively to the respective output offsets, and computing respective modulos of the respective binary-shifted values; and

combining the sub-block error correction terms to determine the error correction term.

31. (original) The method· according to claim 30, wherein the input sub-blocks have respective input offsets within the input block, and wherein

determining the respective sub-block error correction terms for the input sub-blocks comprises binary-shifting a value of each of the input sub-blocks responsively to the respective input offsets, and computing respective modulos of the respective binary-shifted values.

32. (original) A protocol processor, comprising:

a receiving circuit, adapted to receive an input block of data together with a modulo-based input error detection code associated with the input block, the input block comprising a plurality of sub-blocks;

a parser, adapted to select a subset of the sub-blocks to be included in an output block;

a correction term calculator, adapted to determine an error correction term based on the selected subset; and

an aggregator, adapted to concatenate the selected subset of the sub-blocks together with the input error detection code and the error correction term to generate an output block for conveyance to a destination processor.

33. (original) The processor according to claim 32, wherein the error correction term is equal to a binary difference between the input error detection code and an output error detection code of the output block.

34. (original) The processor according to claim 32, wherein the parser is adapted to determine an order of the sub-blocks in the output block, and wherein the correction term calculator is adapted to determine the error correction term responsively to the order.

35. (original) The processor according to claim 32, wherein the correction term calculator is adapted to process data in each of the sub-blocks so as to compute respective sub-block error detection codes.

36. (original) The processor according to claim 35, wherein the correction term calculator is adapted to process the data in each of the sub-blocks by taking a modulo of the data.

37. (original) The processor according to claim 36, wherein the correction term calculator is adapted to take the modulo by computing the modulo with respect to a predetermined polynomial, so as to determine a cyclic redundancy code (CRC) of the sub-block.

38. (original) The processor according to claim 37, wherein the correction term calculator is adapted to compute the modulo using the predetermined polynomial that was applied in computing the input error detection code.

39. (original) The processor according to claim 35, wherein the sub-blocks have respective input offsets within the input block, and wherein the correction term calculator is adapted to determine the error correction term by determining respective sub-block error correction terms for the sub-blocks, responsively to the respective sub-block error detection codes and input offsets, and combining the sub-block error correction terms to determine the error correction term.

40. (original) The processor according to claim 39, wherein the correction term calculator is adapted to determine the respective sub-block error correction terms by binary-shifting the respective sub-block error detection codes by respective numbers of bits equal to the respective input offsets, and computing respective modulos of the respective shifted values.

41. (original) The processor according to claim 39, wherein the sub-blocks in the selected subset have respective output offsets within the output block, and wherein the correction term calculator is adapted to determine the respective sub-block error correction terms for the sub-blocks in the subset by determining the respective sub-block error correction terms responsively to the respective sub-block error detection codes, input offsets, and output offsets.

42. (original) The processor according to claim 41, wherein the correction term calculator is adapted to determine the respective sub-block error corrections terms by multiplying the respective sub-block error detections codes by respective sums of (a) 2 raised to a power of a value of the respective input offsets and (b) 2 raised to a power of a value of the respective output offsets, and computing respective modulos of the respective multiplied values.

43. (original) The processor according to claim 32, wherein the sub-blocks have respective input offsets within the input block, and wherein the correction term calculator is adapted to determine the error correction term by determining respective sub-block error correction terms for the sub-blocks by binary-shifting a value of each of the sub-blocks responsively to the respective input offsets, computing respective modulos of the respective binary-shifted values, and

combining the sub-block error correction terms to determine the error correction term.

44. (original) The processor according to claim 43, wherein the sub-blocks in the selected subset have respective output offsets within the output block, and wherein the correction term calculator is adapted to determine the respective sub-block error correction terms for the sub-blocks in the subset by binary-shifting a value of each of the sub-blocks responsively to the respective output offsets, and computing respective modulos of the respective binary-shifted values.

45. (withdrawn) A data receiver, comprising:

a receiving circuit, adapted to receive a block of data having a modulo-based input error detection code and an error correction term appended thereto; and

an error detection circuit, which is coupled to compute an output error detection code of the block received by the receiving circuit, to combine the input error detection code and the error correction term to produce a modified error detection code, and to compare the calculated error detection code to the modified error detection code so as to detect an error in the block.

46. (withdrawn) The receiver according to claim 45, wherein the error detection circuit is adapted to combine the appended error detection code and the error correction term by applying an XOR operation.

47. (original) A computer system, comprising:

a protocol processor, which comprises:

a receiving circuit, adapted to receive an input block of data together with a modulo-based input error detection code associated with the input block, the input block comprising a plurality of sub-blocks;

a parser, adapted to select a subset of the sub-blocks to be included in an output block;

a correction term calculator, adapted to determine an error correction term based on the selected subset; and

an aggregator, adapted to concatenate the selected subset of the sub-blocks together with the input error detection code and the error correction term to generate an output block; and

a destination processor, which is coupled to receive the output block from the protocol processor and to verify the data in the output block responsively to the input error detection code and the error correction term.

48. (original) The system according to claim 47, wherein the destination processor is adapted to determine whether to accept or reject the output block by computing an output error detection code of the output block, and comparing the output error detection code to the input error detection code and the error correction term.

49. (original) The system according to claim 48, wherein the destination processor is adapted to compare the output error detection code by applying an XOR operation to the input error detection code and the error correction term.

50. (original) The system according to claim 47, wherein the error correction term is equal to a binary difference between the input error detection code and an output error detection code of the output block.

51. (original) The system according to claim 47, wherein the parser is adapted to determine an order of the sub-blocks in the output block, and wherein the correction term calculator is adapted to determine the error correction term responsively to the order.

52. (original) The system according to claim 47, wherein the correction term calculator is adapted to process data in each of the sub-blocks so as to compute respective sub-block error detection codes.

53. (original) A protocol processor, comprising:

a receiving circuit, adapted to receive an input block of data together with a modulo-based input error detection code associated with the input block, the input block comprising a plurality of input sub-blocks;

a parser, adapted to generate one or more protocol-related sub-blocks to be incorporated together with the input sub-blocks in a specified order in an output block for conveyance to a destination processor;

a code calculator, adapted to determine an error correction term based on the specified order of the protocol-related sub-blocks and the input sub-block, and to determine a modulo-based output error detection code responsively to the input error detection code and the error correction term; and

an aggregator, adapted to concatenate the protocol-related sub-blocks, the

input sub-blocks and the output error detection code to generate an output block for conveyance to a destination processor.

54. (original) The processor according to claim 53, wherein the error correction term is equal to a binary difference between the input error detection code and the output error detection code.

55. (original) The processor according to claim 53, wherein at least one of the protocol-related sub-blocks comprises a header, a marker, or padding.

56. (original) The processor according to claim 53, wherein the code calculator is adapted to determine the output error detection code by applying an XOR operation to the input error detection code and the error correction term.

57. (original) The processor according to claim 53, wherein the code calculator is adapted to determine the error correction term by processing the data in each of the pro tocol-related sub-blocks and the input sub-blocks so as to compute respective output sub-block error detection codes.

58. The processor according to claim 57, wherein the code calculator is adapted to process the data in each of the protocol-related sub-blocks and the input sub-blocks by taking a modulo of the data.

59. (original) The processor according to claim 58, wherein the code calculator is adapted to take the modulo by computing the modulo with respect to

a predetermined polynomial, so as to determine a cyclic redundancy code (CRC).

60. (original) The processor according to claim 59, wherein the code calculator is adapted to compute the modulo by using the predetermined polynomial that was applied in computing the input error detection code.

61. (original) The processor according to claim 57, wherein the code calculator is adapted to determine the correction term by:

determining respective output offsets of the protocol-related sub-blocks and the input sub-blocks within the output block,

determining respective sub-block error correction terms for the protocol-related sub-blocks and the input sub-blocks, responsively to the respective sub-block error detection codes and output offsets, and

combining the sub-block error correction terms to determine the error correction term.

62. (original) The processor according to claim 61, wherein the code calculator is adapted to determine the respective sub-block error correction terms by binary-shifting the respective sub-block error detection codes by respective numbers of bits equal to the respective output offsets, and computing respective modulos of the respective shifted values.

63. (original) The processor according to claim 61, wherein the input sub-blocks have respective input offsets within the input block, and wherein the code calculator is adapted to determine the respective sub-block error correction terms

for the input sub-blocks by determining the respective sub-block error correction terms responsively to the respective sub-block error detection codes, input offsets, and output offsets.

64. (original) The processor according to claim 63, wherein the code calculator is adapted to determine the respective sub-block error corrections terms by multiplying the respective sub-block error detections codes by respective sums of (a) 2 raised to a power of a value of the respective input offsets and (b) 2 raised to a power of a value of the respective output offsets, and computing respective modulos of the respective multiplied values.

65. (original) The processor according to claim 53, wherein the code calculator is adapted to determine the error correction term by:

determining respective output offsets of the protocol-related sub-blocks and the input sub-blocks within the output block, responsively to the specified order,

determining respective sub-block error correction terms for the protocol-related sub-blocks and the input sub-blocks by binary-shifting a value of each of the protocol-related sub-blocks and the input sub-blocks responsively to the respective output offsets, and computing respective modulos of the respective binary-shifted values, and

combining the sub-block error correction terms to determine the error correction term.

66. (original) The processor according to claim 65, wherein the input sub-

blocks have respective input offsets within the input block, and wherein the code calculator is adapted to determine the respective sub-block error correction terms for the input sub-blocks by binary-shifting a value of each of the input sub-blocks responsively to the respective input offsets, and computing respective modulos of the respective binary-shifted values.

67. (original) A computer system, comprising:

a source processor, which is adapted to generate an input block of data, comprising a plurality of input sub-blocks, and to generate a modulo-based input error detection code for the input block; and

a protocol processor, coupled to receive the input block together with the modulo-based input error detection code, and comprising:

a parser, adapted to generate one or more protocol-related sub-blocks to be incorporated together with ·the input sub-blocks in a specified order in an output block for conveyance to a destination processor;

a code calculator, adapted to determine an error correction term based on the specified order of the protocol-related sub-blocks and the input sub-blocks, and to determine a modulo-based output error detection code responsively to the input error detection code and the error correction term; and

an aggregator, adapted to concatenate the protocol-related sub-blocks, the input sub-blocks and the output error detection code to generate an output block for conveyance to a destination processor.

68. (original) The system according to claim 67, wherein the error

correction term is equal to a binary difference between the input error detection code and the output error detection code.

69. (original) The system according to claim 67, wherein the code calculator is adapted to determine the output error detection code by applying an XOR operation to the input error detection code and the error correction term.

70. (original) The system according to claim 67, wherein the code calculator is adapted to determine the error correction term by processing data in each of the protocol-related sub-blocks and the input sub-blocks so as to compute respective output sub-block error detection codes.

71. (withdrawn) A computer software product for receiving data, the product comprising a computer-readable medium in which program instructions are stored, which instructions, when read by a computer, cause the computer to receive a block of data having a modulo-based input error detection code and an error correction term appended thereto, to compute an output error detection code of the block, to combine the input error detection code and the error correction term to produce a modified error detection code, and to compare the calculated error detection code to the modified error detection code so as to detect an error in the block.

72. (withdrawn) The product according to claim 71, wherein the instructions cause the computer to combine the appended 20 error detection code and the error correction term by applying an XOR operation.

73. (new) A machine-readable storage having stored thereon, a computer program having at least one code section for data verification, the at least one code section being executable by a machine for causing the machine to perform steps comprising:

receiving an input block of data together with a modulo-based input error detection code associated with the input block, the input block comprising a plurality of input sub-blocks;

generate one or more protocol-related sub-blocks to be incorporated together with the input sub-blocks in a specified order in an output block for conveyance to a destination processor;

determining an error correction term based on the specified order of the protocol-related sub-blocks and the input sub-blocks;

determining a modulo-based output error detection code responsively to the input error detection code and the error correction term; and

concatenating the protocol-related sub-blocks, the input sub-blocks and the output error detection code to generate an output block for conveyance to a destination processor with the input block, the input block.